Efficient Continuous Algorithms for Combinatorial Optimization

Anil P. Kamath
Department of Computer Science
Stanford University
Stanford, CA 94305
(kamath@cs.stanford.edu)
(http://theory.stanford.edu/people/kamath/home.html)

March 8, 1995

1 Continuous vs. Combinatorial

Combinatorial: Spanning tree

Matching

Single Commodity Flow

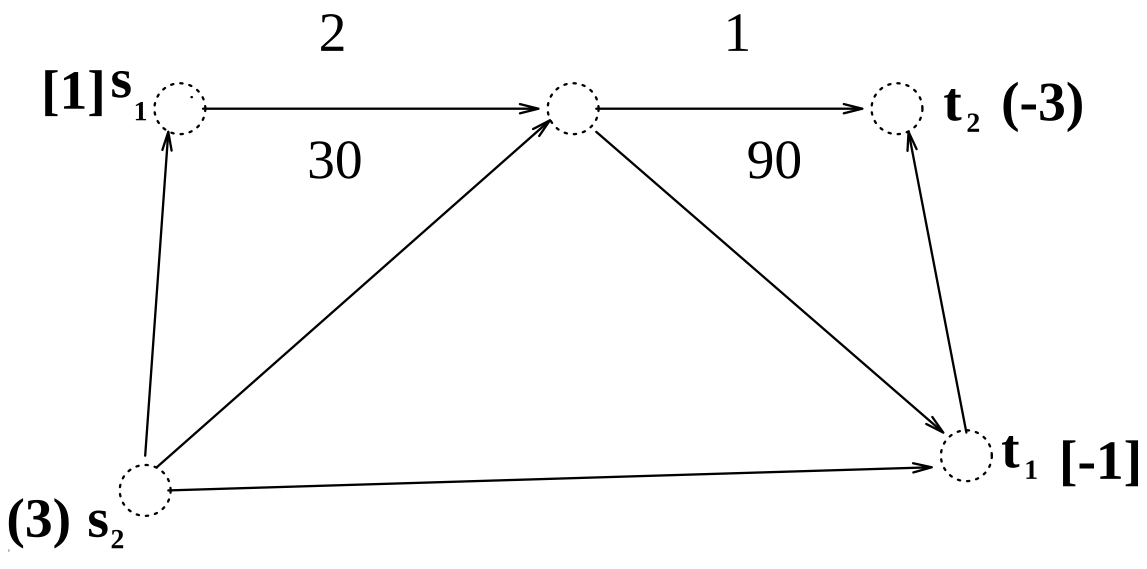
Multicommodity Flow

Fleet Scheduling Continuous

2 Minimum Cost MultiCommodity Problem

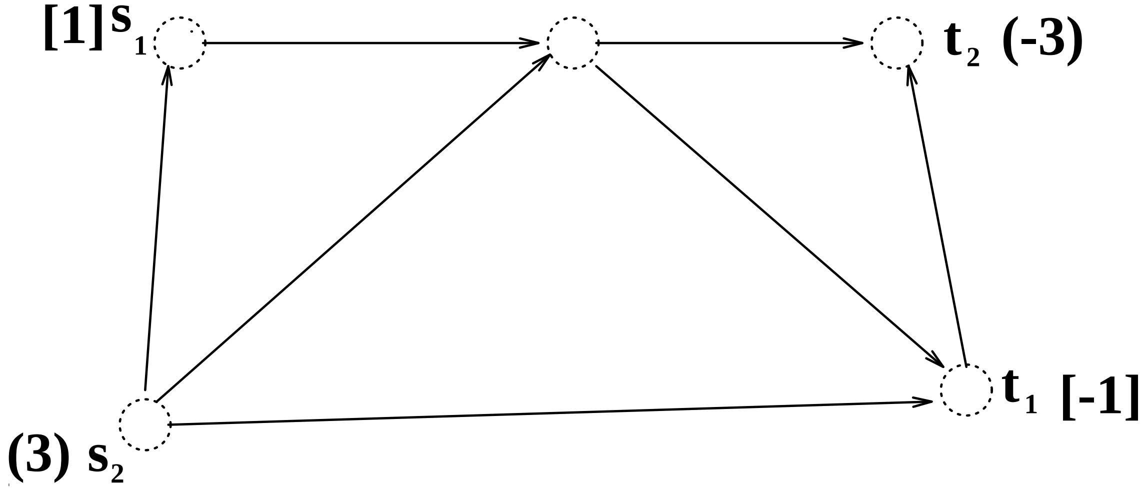
- commodity i
  source s_i, sink t_i, demand d_i

- edge vw
  capacity $u(vw)$, cost $c(vw)$



3 Minimum Cost MultiCommodity Flow

- capacity constraints
- conservation constraints
- budget constraints



4 Applications

- VLSI design
- Network design
- Routing
- Fleet Scheduling
- Crew Scheduling

5 Outline of Talk

•Linear Programming •Quadratic Programming	Interior-Point
•Approximation Methods •Distributed Techniques	Potential-Red.
•Routing in ATM Networks	Lagrange relax.

6 Linear Programming Formulation

<u>Capacity Constraints</u>:

$$\begin{aligned} f^{(i)}(e) &\geq 0 \qquad \forall e \in \mathcal{E}\,,\, i \in \mathcal{K} \\ \sum_{i \in \mathcal{K}} f^{(i)}(e) &\leq u(e)\ \forall e \in \mathcal{E} \end{aligned}$$

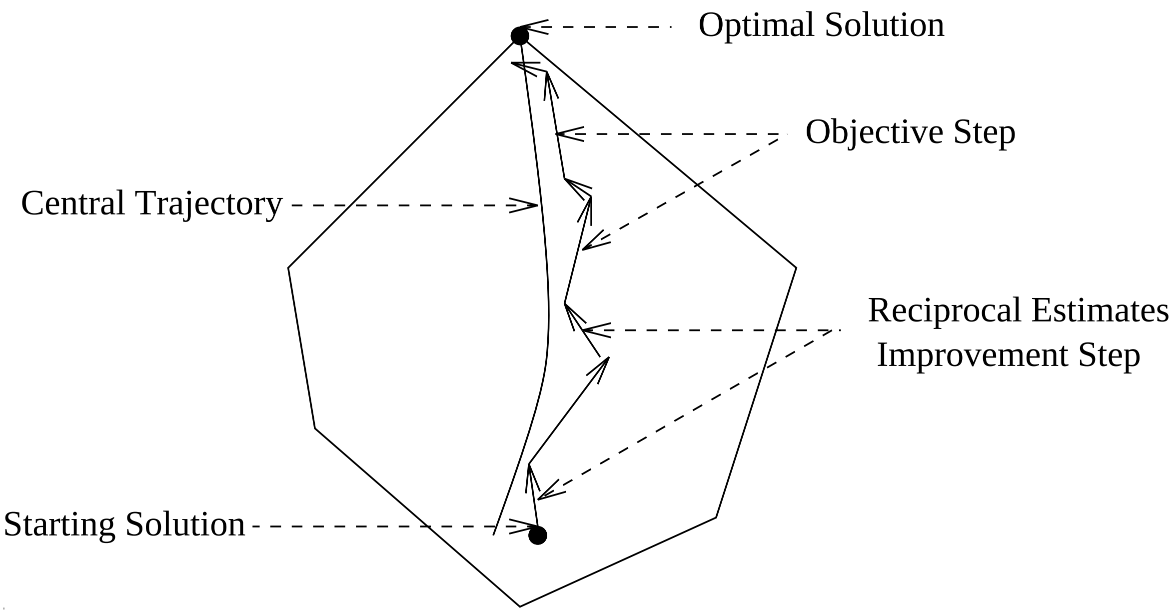
<u>Conservation Constraints</u>:

$$\sum_{wv \in \mathcal{E}} f^{(i)}(wv) - \sum_{vw \in \mathcal{E}} f^{(i)}(vw) = \begin{cases} 0 & v \neq s_i; v \neq t_i \\ -d_i & v = s_i \\ d_i & v = t_i \end{cases} .$$

The objective:
Find *feasible* flow with *minimum* total cost.

$$\min \sum_{i \in \mathcal{K}} \sum_{e \in \mathcal{E}} c^{(i)}(e) f^{(i)}(e).$$

7 Approximate Dual Projective Method



Central path:

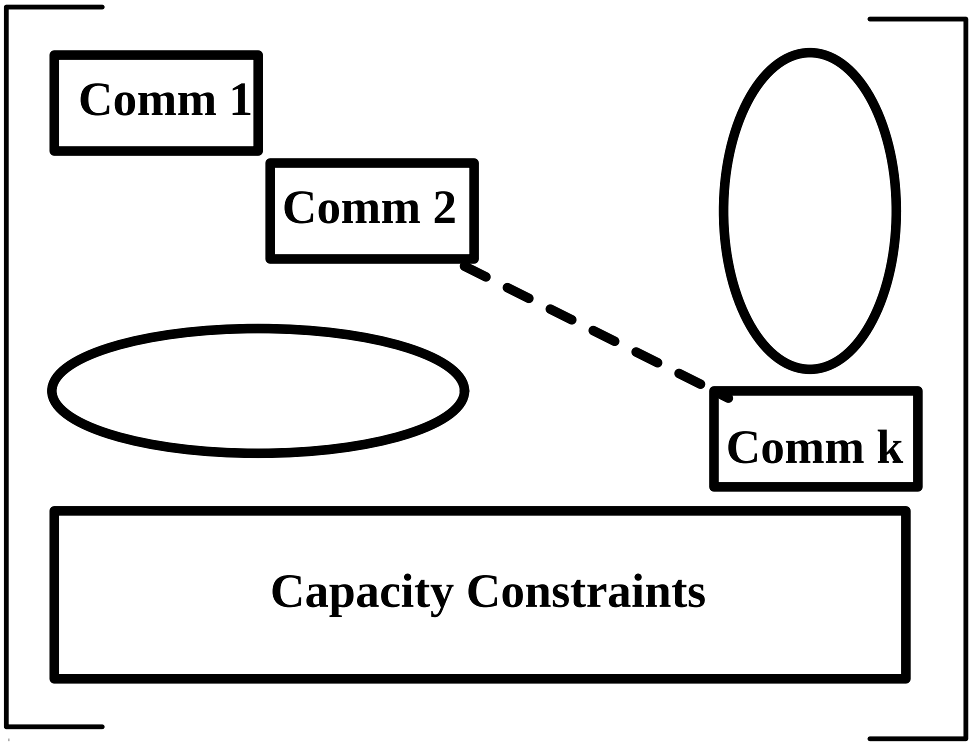
$$\rho(\mu) \quad \text{minimizes} \quad (\text{objective}) - \mu \cdot (\text{barrier})$$

Computationally intensive step

$$AD^2A^Tx \ = \ b$$

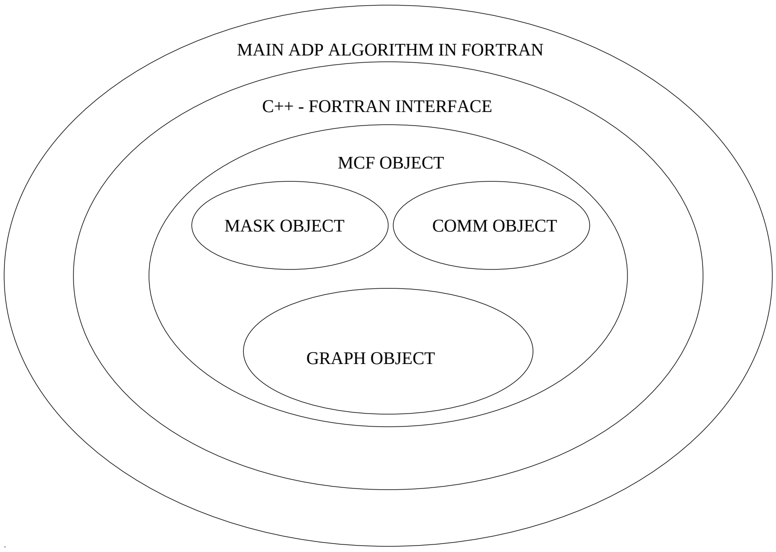
A is the constraint matrix of MCF
D is a diagonal scaling matrix.

8 Efficient Implementation



- Direct Factorization: Dense, Inefficient
- Preconditioned Conjugate Gradient
  Each CG step - matrix-vector multiply
- Theoretical complexity/iteration: $O(k^2mn)$
  Implementation Tricks:
- efficient preconditioning
- store constraint matrix implicitly -
  single copy of incidence matrix
- capacity constraints *not stored*

9 Object-Oriented Software



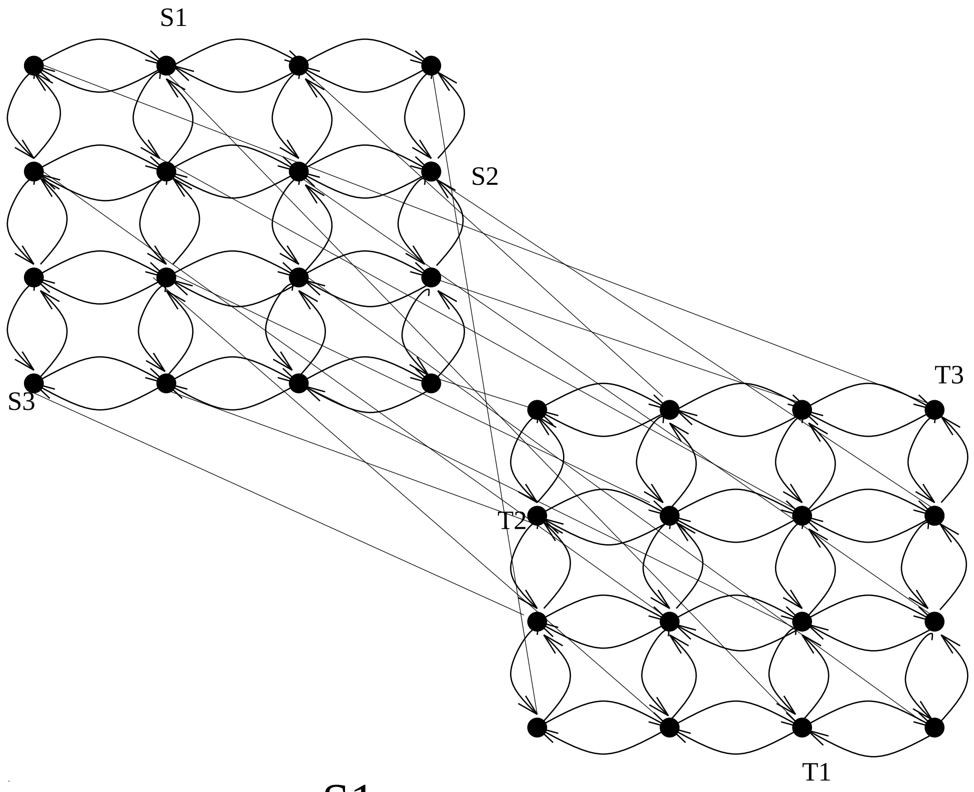
- reduction in storage:
  store single copy of incidence matrix

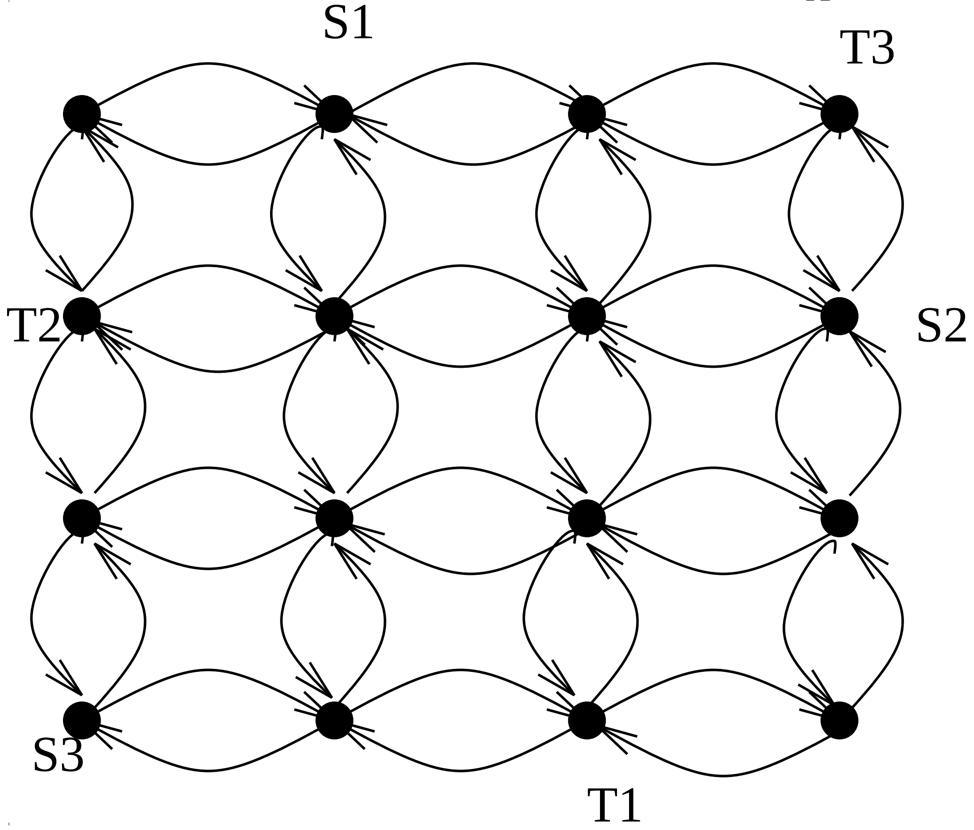
- ease of vectorization:
  edge corresponds to a vector of length $2k$ in the expanded constraint matrix

- reduction in data access:
  accesses to fetch constraint matrix reduced.

- software reuse:
  main ADP algorithm oblivious to the matrix representation

10 Generators





11 Computational Results

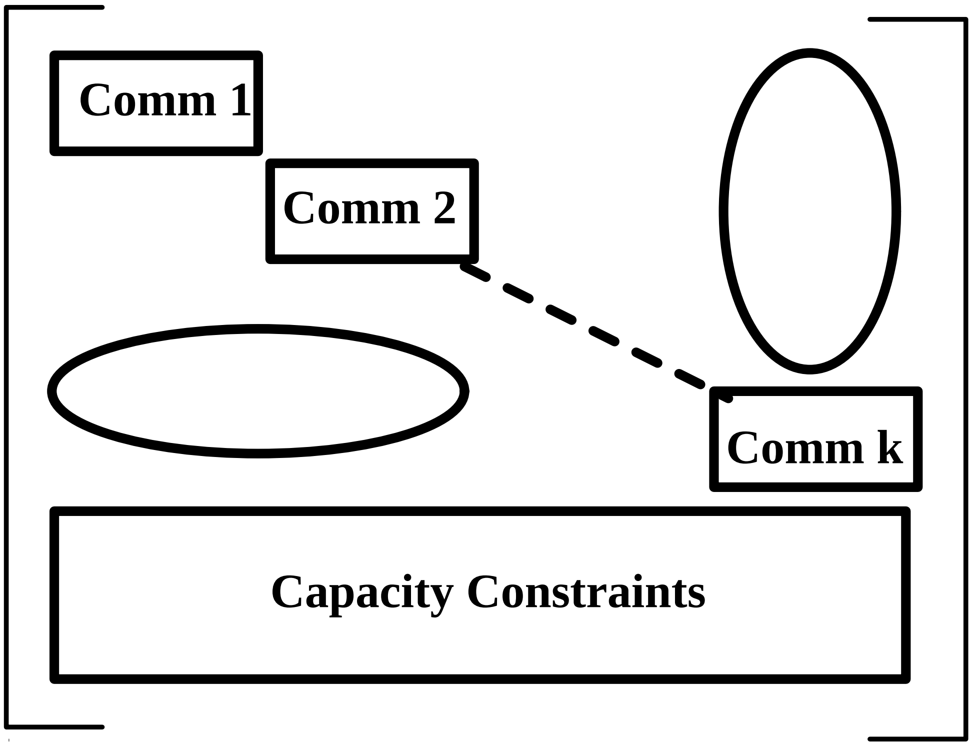
Problem	Rows in A	Columns in A	ADP Time	CPLEX-NETOPT Time
16.8.4	17664	47360	283s	685s
16.8.8	25856	85248	505s	1253s
16.8.16	42240	161024	1441s	2300s
16.16.4	35584	96000	313s	2933s
16.16.8	51968	172800	1620s	5582s
16.16.16	84736	326400	6363s	11000s
16.32.4	71424	193280	1580s	12807s
16.32.8	104192	347904	10596s	23553s
16.32.16	169728	657152	19834s	44001s

Table 1: Performance of ADP on *rmfgen* problems

Grid Size	Comm.	Rows	Columns	ADP Time	CPLEX NETOPT
20	5	3520	9120	147s	408s
50	5	22300	58800	1740s	67956s
100	5	89600	237600	13102s	$>$ 556803s

Table 2: Performance of ADP on *Mgrid* problems

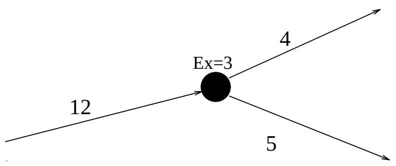
12 Problems with Linear Programming Approach



- different types of constraints
- absence of good starting point

13 Multicommodity Flow as Quadratic Programming

Excess at a node:



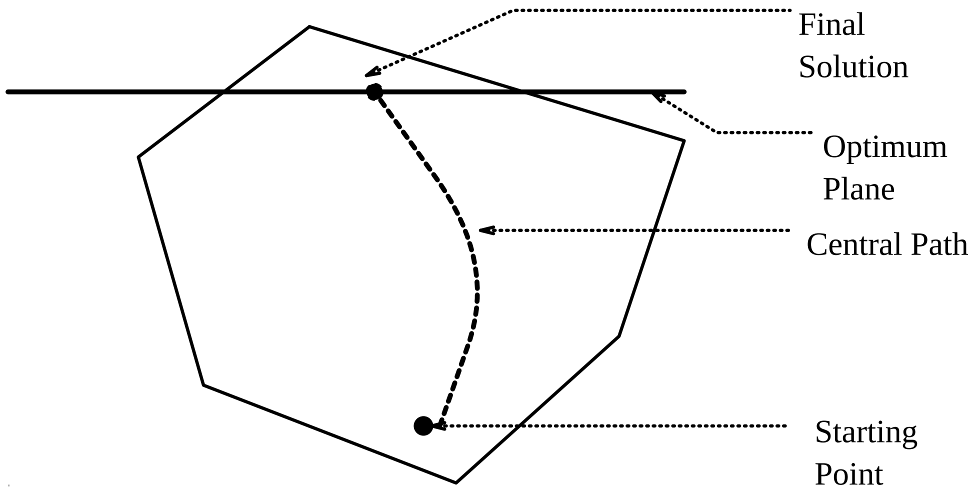
$$\text{Goal:} \quad \text{minimize} \quad \sum_v (Excess(v))^2$$

Solution with value 0 $\iff$ *feasible* flow.

0 is minimal value of objective function.

Keep linear *capacity* constraints.
No *conservation* constraints.

14 Interior Point for Quadratic Programming



Using the Monteiro - Adler algorithm
Central path:

$$\rho(\mu) \quad \text{minimizes} \quad (\text{objective}) - \mu \cdot (\text{barrier})$$

$$\mu \to 0 \quad \Longrightarrow \quad \rho(\mu) \to \text{solution}$$

Iteration i:

$$x^i \approx \rho(\mu^i)$$

$$\mu^{i+1} = (1 - \delta)\mu^i$$

Use Newton step to compute x^{i+1} from x^i.

15 Computation of Each Iteration

Bottleneck: given $w \in R^m$ find $v \in R^m$ such that

$$\left((D^s)^{-1} + \sum_{l \le k} \left(D^l + B^T B\right)^{-1} \right) v = w$$

where D^i are all diagonal.

1. Use conjugate gradient to solve:

$$\begin{pmatrix} I & I & \cdots & I \\ -D^s & D^1 + B^T B & & \\ \vdots & & \ddots & \\ -D^s & & & D^k + B^T B \end{pmatrix} \begin{pmatrix} w_s \\ w_1 \\ \vdots \\ w_k \end{pmatrix} = \begin{pmatrix} w \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

2. Use matrix inversions and rank one updates.

16 Starting Point

Analytical center of polytope :
point that maximizes barrier function.

Standard approach: transform to a problem with *known* analytical center.
Problem: starting point is exponential.

$$\log \mu_{\text{start}} = O\left(nk\right)$$

Our formulation: Simple Polytope
Analytical center:

$$f_l(e) = f_s(e) = \frac{u(e)}{k+1} \quad .$$

$$\log \mu_{\text{start}} = O\left(\log nDU\right)$$

D — biggest demand, U — biggest capacity.

17 Complexity

	previous	our algo.
One iteration	$k^{2.5}n^{2.5}$	k^2nm or $m^{2.5}$
Step length	same	$1/\sqrt{mk}$
$\log \mu_{\text{start}}$	$\underline{O(nk)}$	$\underline{O(\log nDU)}$
$-\log \mu_{\text{end}}$	same	$\underline{O(\log \epsilon^{-1} m)}$

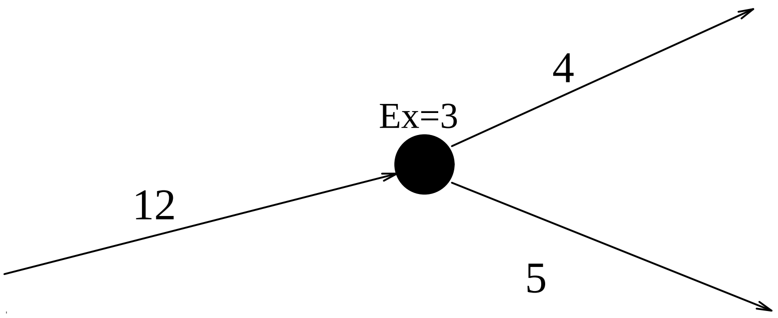
Complexity:

$$\frac{\text{one iteration}}{\text{step length}} \cdot \Big(\log \mu_{\text{start}} - \log \mu_{\text{end}}\Big)$$

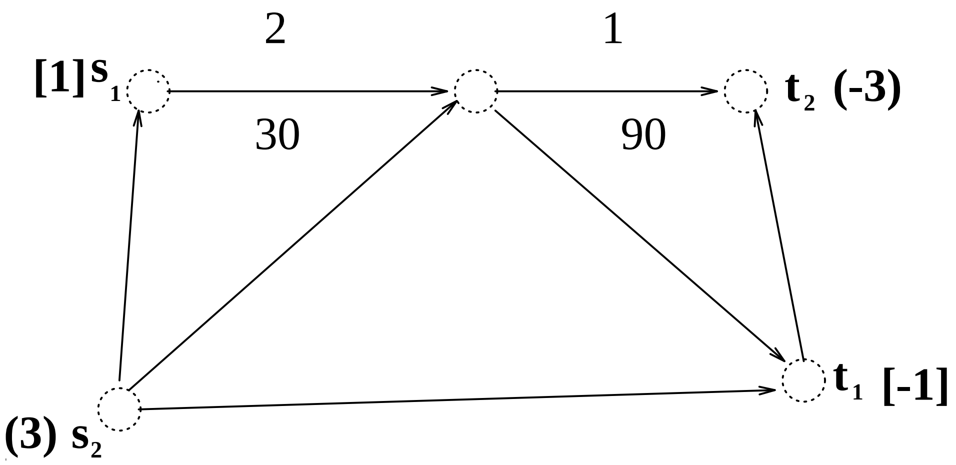
18 Pre-Flow and Excesses

pre-flow based algorithm [GT90]

- obeys capacity
- *does not* obey conservation

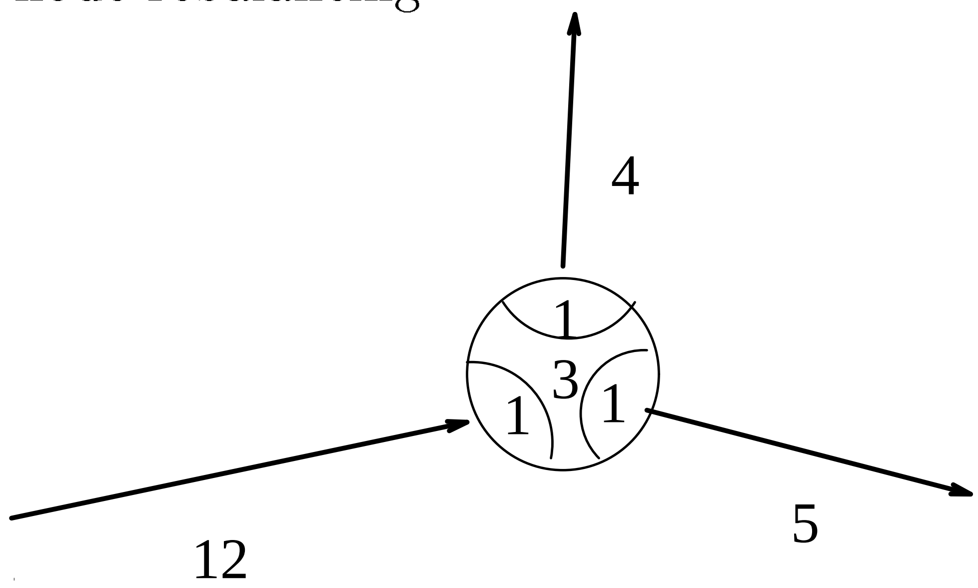


Goal: To Minimize the Excess at each Node

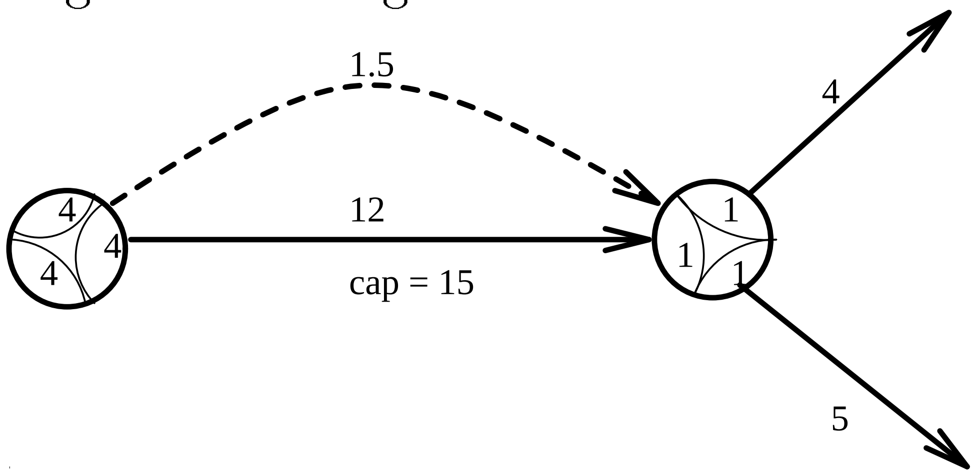


19 Rebalancing

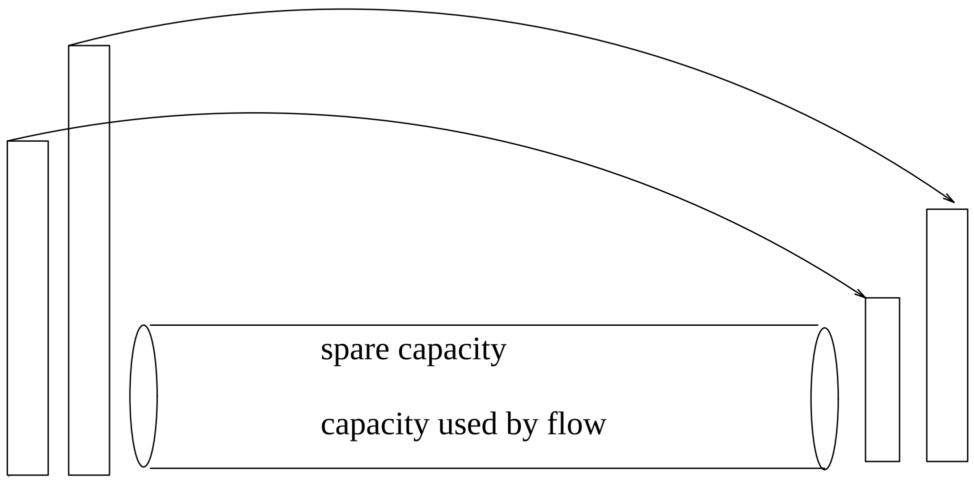
- node rebalancing



- edge rebalancing



20 Edge-Rebalancing



How to share capacity among commodities?

- without costs - objective to decrease the excesses at every node
  minimize some convex function of excesses

- with costs - also reduce cost of flow
  add cost function to the objective?

21 Potential Function

Penalty for excess

$$\Phi_1 = \sum_{i, vw \in E \text{ or } wv \in E} \exp\left(\alpha \frac{Ex_i(f, vw)}{d_i^*}\right),$$

Penalty for cost

$$\Phi_2 = \frac{1}{B} \sum_{e \in E} c(e) f(e).$$

To measure progress use potential function:

$$\Psi = \log \frac{\Phi_1}{2mk} + \frac{\gamma}{1+\delta} \Phi_2.$$

But the function is *not edge separable.*

Define new objective:

$$\Phi = \Phi_1 + \gamma \Phi_2 \tilde{\Phi}_1$$

Φ : separable - minimized on per edge basis

- algorithm uses Φ for reduction
- proof uses Ψ for convergence

22 Complexity

Relaxation of *conservation* constraints

- (without costs)
  $[AL94, our\ method] = O^*(kmn^2)$

- (with costs)

$$\begin{aligned} [our\ method] &= O^*(kmn^2)\ (deterministic) \\ &= O^*(km^3)\ \ (distributed) \end{aligned}$$

Relaxation of *Capacity* constraints

- (without costs)
  $[LMPSTT91] + [R95] = O^*(kmn)$

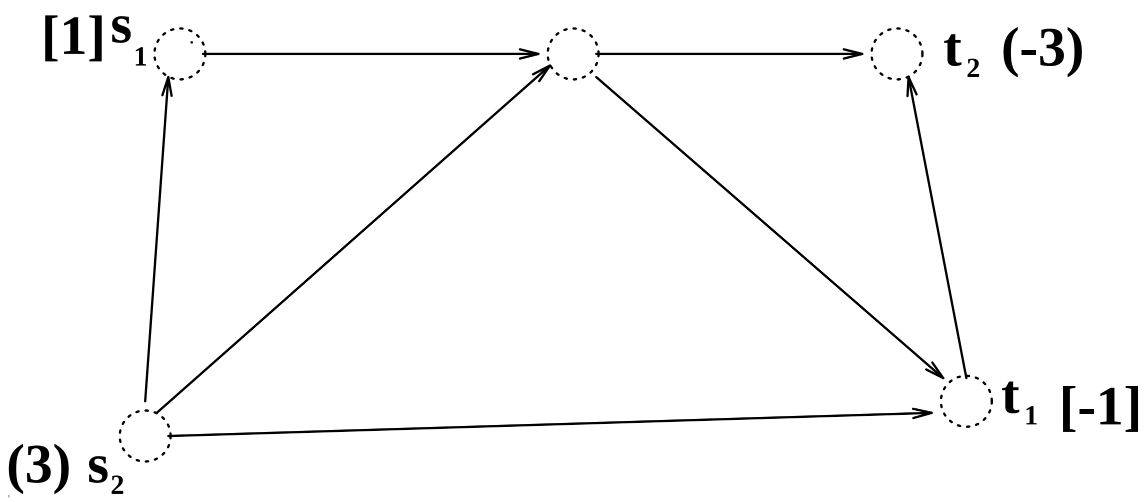
- (with costs)

$$\begin{aligned} [PST91] &= O^*(k^2m^2)\ \ (deterministic) \\ &= O^*(km^2)\ \ (randomized) \end{aligned}$$

[Interior-Point] $= O^*(k^{2.5}m^{1.5}n)$

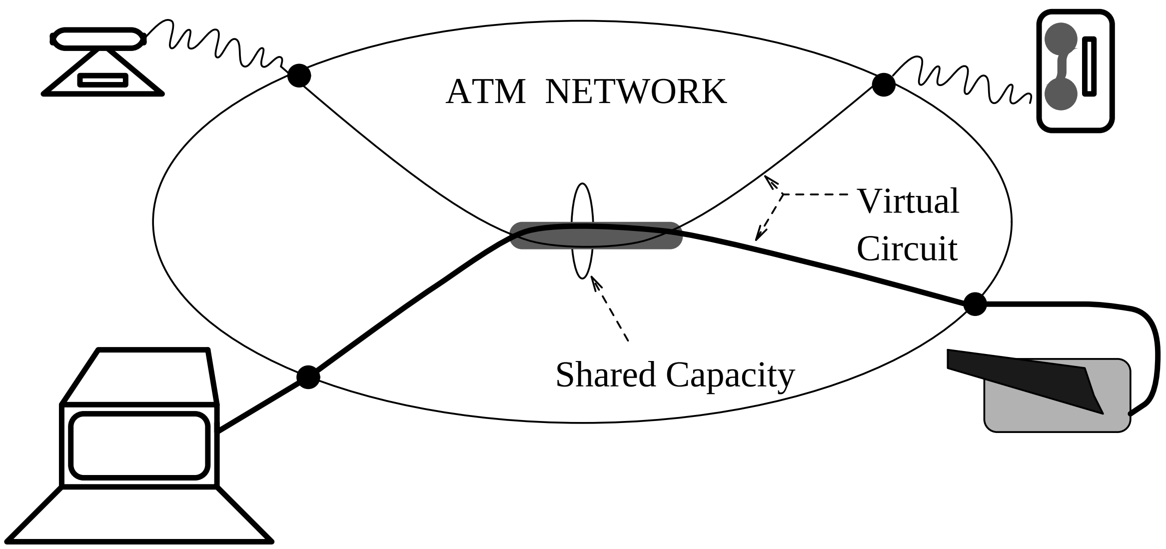
23 Routing: Integral Multi-Commodity Flow

- integral units
- single-path



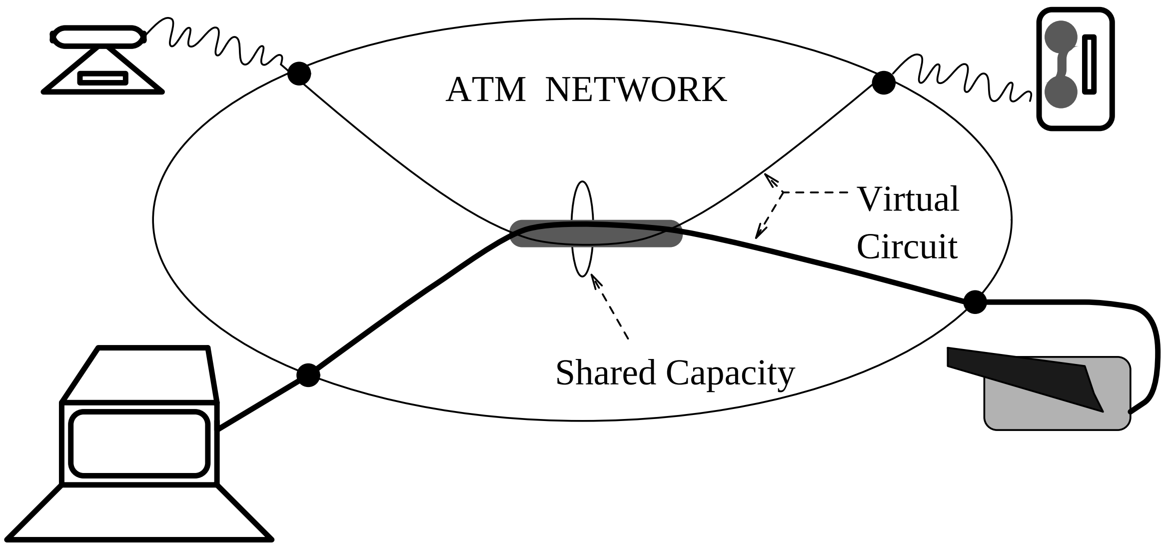
24 High-Speed Networks

- High bandwidth consumer services
  - video-on-demand
  - Teleconferencing
  - mult-media applications
- Network resources(bandwidth, buffer spaces) shared.



25 Resource Reservation

- Quality of Service guarantees required $\Longrightarrow$ need *reservation* of resources.
- limited resources - need for admission control.
- Need to allocate bandwidth along some path from source to destination.
- resources allocated in terms of Virtual Circuits.



26 Off-line Routing: Integral Multi-Commodity Flow

Off-line Routing problem: NP-complete

solve MCF problem + use randomized rounding [Raghavan/Thompson]

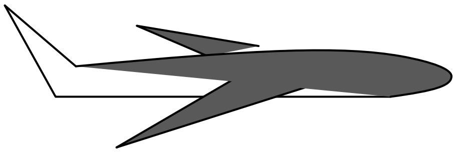
Direct approximation methods:[PST91]
Edge cost - *exponential in congestion*

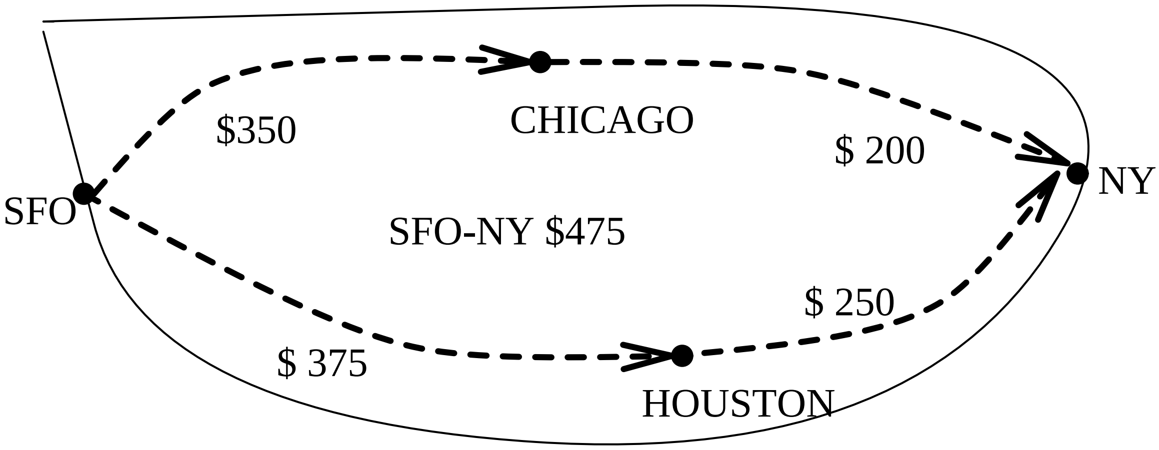
Start with some solution:
LOOP:

- pick a commodity
- reroute onto cheaper path
- update edge costs

---

27 On-line Routing and Admission Control





On-line Single-Path Multi-commodity Flow
Two related questions:

- whom to accept vs.
  whom to reject?: Admission Control
- which route to use?: Routing

Goal: Maximize profit

28 Competitive Ratio

How to evaluate on-line routing strategies?

- Problem: "Bad" input sequence for any strategy
- use competitive ratio

$$\boxed{\text{Worst over all sequences } \frac{\text{Off-line Performance}}{\text{On-line Performance}}}$$

- Small competitive ratio $\Longleftrightarrow$ near-optimum performance
- Performance Measure
  - Maximum congestion on a link [AAFPW-93]
  - Throughput(#ckts or bandwidth routed) [AAP-93]

29 On-line Approach

- Use exponential costs to represent congestion
- Route on shortest path w.r.t. these costs.
- Rejections: reject if cost “too high”
- Maintain close to optimum solution

30 Theoretically Optimum Algorithms

Maximizing Throughput:[AAP-93]

- Define cost of edge e

$$cost(e) = \int_t (\mu^{\lambda_e(t)} - 1)dt$$

- $\lambda_e(t)$ = congestion of edge e at time t
  $\mu = 2$ (max-duration[T])(max-hops[L]) + 1.
- each VC has an associated profit.
- route along shortest path w.r.t. this cost.
- Reject if length of shortest path exceeds profit.

$\boxed{O(\log LT) \text{ - competitive w.r.t. profit}}$

31 Are these Algorithms Practical?

- Geared towards the worst case.
- Centralized
- needs duration
- Polylogarithmic bound on performance is unsatisfactory.

Solutions?

- optimize average case performance
- distributed implementation
- work with distributions on holding time
- prove good average-case performance guarantees

32 Random Duration

Average case Analysis:
Known Durations:

- Durations random with certain properties on distribution, but exact duration known at arrival: $O(\log L)$ competitive.

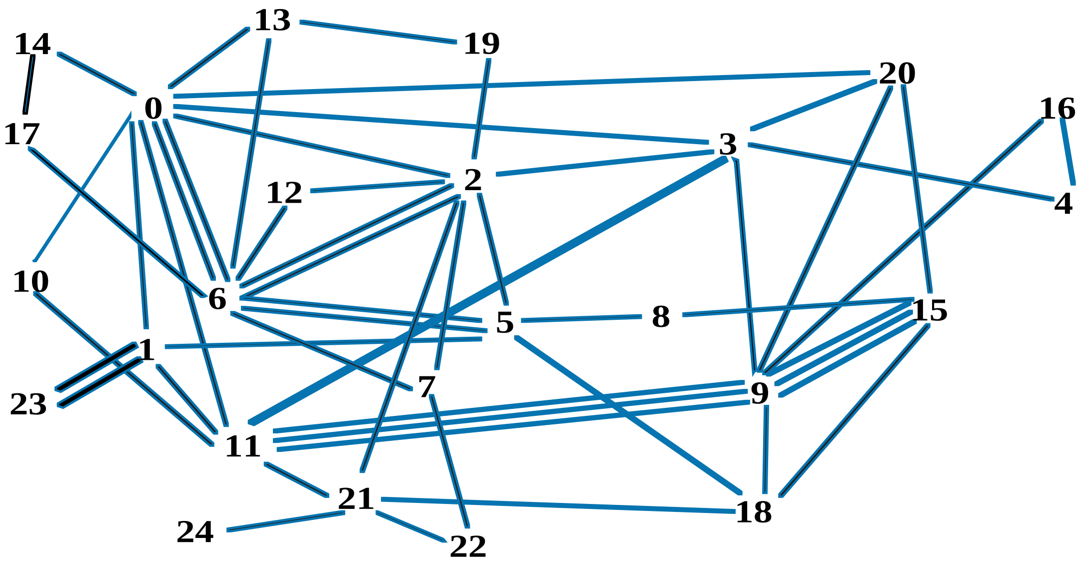
Unknown durations:

- A priori knowledge of holding time distribution is sufficient

- $O(\log LT)$-competitive ratio against adversary that also knows only distribution on holding times

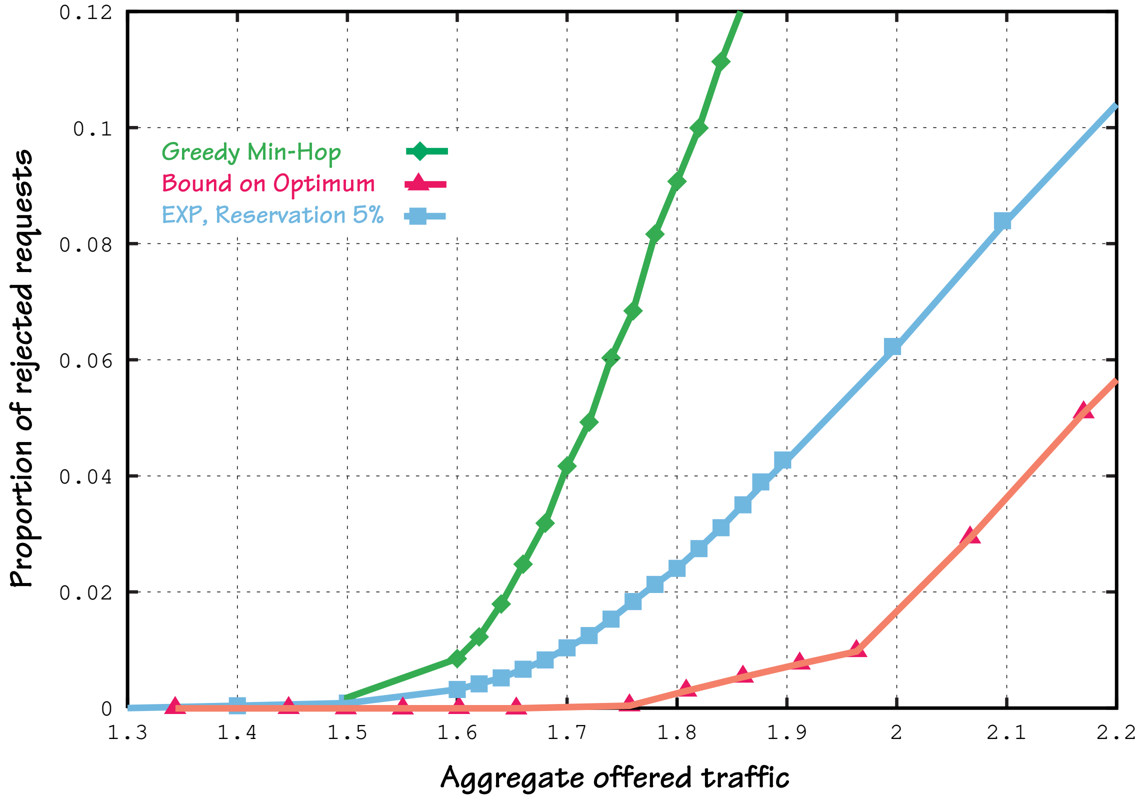
- $O(\log L \log\log L)$-competitive ratio for exponentially distributed holding times.

33 Topology



- Commercial network Topology
  Each link: SONET OC-3 (155Mbits)
- traffic matrix represents real usage
- Holding times:exponential, mean 30 minutes.
- Objective: Maximize calls accepted

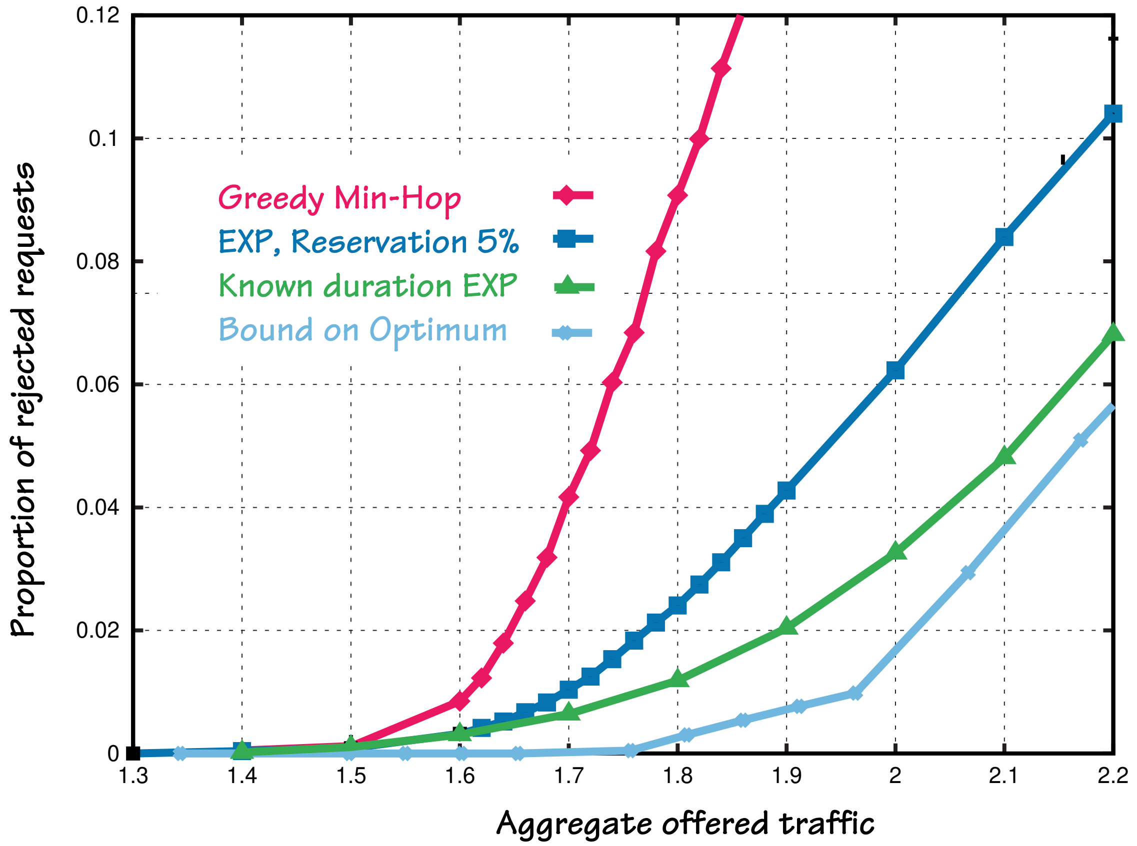
34 EXP vs. Optimum vs. Greedy



Comments:

- Optimum bound: Fractional LP
- Target Rate 2% rejection 88% vs 81%
- Target Rate 10% rejection 88% vs 73%

Does it help to know Duration?



- Movies, TV classes, Video Conferences: Duration known at arrival
- Information about Duration can be used
- Significant improvement over techniques that do not use duration

35 Conclusions

- efficient algorithms based on LP and QP for MCF
- fast and distributed approximation algorithms
- applications to on-line routing of VCs in ATM Networks